DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Proposal for Master's Thesis in Automotive Software Engineering

Hardware-in-the-Loop Test Setup for a Generalistic Approach to Machine-Learning-Based Schedulability Analysis

Hardware-in-the-Loop Testsetup eines generalistischen Ansatzes zur Schedulabilityanalyse basierend auf maschinellem Lernen

Author:	Octavio Ivan Delgadillo Ruiz
Supervisor:	Prof. Dr. Uwe Baumgarten
Advisor:	Bernhard Blieninger, M. Sc.
Registration Date:	15.06.2020

Introduction

One of the biggest trends in the automotive industry is the development of autonomous vehicles that free users from the driving tasks, and so increase their comfort and safety. To be sustainable, this should be combined with another big trend: efficient resource utilization, which is motivated by environmental protection and especially cost reduction, as the automotive sector is primarily cost driven. Using computers efficiently is therefore crucial for this industry, since their relevance is constantly increasing with the grade of automation.

Modern cars include dozens of small computers, a number that is likely to increase if we consider past trends [Vip+14; Vip18]. These computers, commonly known as electronic control units (ECUs), are embedded devices that often must meet certain constraints to ensure the safety of the tasks performed, such as real-time capabilities [Vip+14]. This is one reason why a lot of devices are integrated to serve a specific purpose. Nevertheless, when so many devices are part of a system, many problems arise, for example, inefficient usage of the devices (since many tasks are only executed in specific situations), reduced fault tolerance in case a system fails, and increased weight and cost of the vehicle due to the high number of ECUs and cables [Vip18]. Hence, the automotive industry seeks alternative approaches that help solve these issues — and, particularly, reduce vehicle costs — while prioritizing vehicle safety [Bur+18].

One such approach is ECU consolidation, which aims to run tasks on a few powerful, multi-purpose ECUs, instead of using lots of single-purpose ECUs. Such consolidation raises other challenges, such as higher computation and safety requirements, especially in case of a software or hardware failure (as many tasks could be blocked by a single failure) and for some safety-critical tasks, where redundancy is required to ensure their correct execution [Mun+17]. To cope with these issues, previous projects at the institute, such as KIA4SM [EKB15] and MaLSAMi[1], have introduced the concept of dynamic task migration, which aims to solve these problems by enabling the execution of tasks on various devices, depending on the state of the system. This permits a re-distribution of the tasks onto different devices. To plan the task distribution, schedulability analysis is performed to ensure all tasks can be executed and meet their real-time constraints. Researchers in these projects have explored machine learning algorithms as an interesting alternative to traditional mathematical approaches for performing schedulability analysis. The mathematical approaches, such as the real-time analysis proposed by Buttazzo in [But11], usually require recurrent calculations that consider the dependencies between tasks and shared resources; this means that in spite of providing reliable calculations, they are often too complex and computation intensive for large task sets or if complex function calls are involved, such as memory allocation or input and output calls [Utz19; NMM19]. Additionally, these approaches also consider worst-case executions and are therefore often too pessimistic. In contrast, the machine-learning-based alternative has shown the potential of being a fast and acceptable approximation of the feasibility of the distribution of the tasks, as demonstrated in previous works [Tai19; Utz19; Bli19].

While the results delivered by previous work seem promising, more evidence is necessary to determine whether machine learning can be used for schedulability analysis on different platforms. To gather such evidence, it is important to test the approach with different pieces of hardware and machine learning algorithms. First evidence has been collected with the hardware setup used in previous work, such as the Hardware-in-the-Loop (HIL) setup used in KIA4SM, and this work shall provide further evidence.

The purpose of this master's thesis is building a Hardware-in-the-Loop setup to evaluate a system using machine learning and an adaptation of the task migration concept to improve reliability, load distribution and system utilization for automotive architectures. While the idea of task migration forms the base for this setup, its full

[1]MaLSAMi project website: https://malsami.github.io/

implementation would require an immense effort and is therefore beyond the scope of the thesis; the concept will be rather explored in the form of the planning and deploying of a new task distribution, unlike the one in KIA4SM. This work shall implement the setup by using the NeuroRobotics Platform (NRP)[2], embedded hardware running FreeRTOS, a simple model car and a SpiNNaker[3] neuromorphic board for the machine learning algorithm. While the HIL setup implemented in this work is based on the one used in KIA4SM, it shall use completely different tools, including the introduction of the neuromorphic architecture for the machine learning tasks. Ideally, this should serve to prove that machine-learning-based schedulability analysis can be adapted to new platforms and perform optimally. To demonstrate this, the system will be tested using time analysis under artificially created critical situations.

Related Work

Research projects KIA4SM and MaLSAMi have explored the possibility of migrating tasks running on a device to another in a real-time capable system (for example, ECUs in a vehicle) under certain conditions. In these works, the migration was divided into two main stages: The first is the migration planning, which determines the hardware that tasks will be migrated to, should the original hardware not be able to fulfill its duty (for example, if there is a failure in that hardware or if the real-time constraints or deadlines would be violated). The second is the execution of the migration, which ensures that corresponding tasks can be migrated from the source device to the target device while keeping their current state. The scope of this thesis only covers the migration planning; hence, the second stage is not relevant for this work.

MaLSAMi analyzed migration planning, with researchers performing schedulability analysis based on machine learning (specifically, on neural networks). Machine learning was picked over traditional mathematical approaches, because those can become too complex for complex tasks and for big task sets, and they often lead to a pessimistic calculation of the system utilization. By predicting the feasibility of a task set using machine learning algorithms, we can obtain potentially faster but less precise results, as demonstrated by previous theses by Taieb [Tai19], Utz [Utz19] and Blieninger [Bli19]. The predictions provided by the machine-learning approach indicate whether a task set is 100% schedulable or not, but they are not completely safe, since false positive predictions may occur. A possible solution to this issue could be punishing false positives heavily and predicting a percentage of success for a task distribution in future states, an idea that has not been implemented yet and will be investigated in this thesis. This approach could be a potentially powerful solution for enabling the execution at run-time of the real-time capable migration planning.

For this purpose, a neuromorphic platform using spiking neural networks is an

[2]NeuroRobotics Platform website: https://www.neurorobotics.net/
[3]SpiNNaker project website: http://apt.cs.manchester.ac.uk/projects/SpiNNaker/

interesting approach, due to the benefits its architecture provides in terms of speed and energy-efficiency when compared to a traditional architecture [BMJ18]. These advantages are present in neuromorphic architectures because they work in a similar way to brains and are therefore well suited for tasks where the brain normally outperforms traditional computers, such as learning. This means a neuromorphic platform could allow for a fast and power-efficient execution of the machine-learning-based schedulability analysis in real time. To explore this possibility, in this thesis the neuromorphic architecture provided by a SpiNNaker board will be used for performing the analysis. An advantage of the SpiNNaker board is that it can be connected with relative ease to a robot body using the NeuroRobotics Platform.

MaLSAMi also produced a data set for training the neural networks used for the schedulability analysis. This data set was generated on real hardware running on GenodeOS, which executed different task sets formed by some dummy tasks. The execution times of the tasks and task sets were measured by performing End-to-End software measurements, which helped provide a better estimation of the system load. The relevance of these training data remains to be proven for different system configurations, which shall be explored for the setup in this thesis.

Additionally, KIA4SM showcased the possibility of building a Hardware-in-the-Loop setup for testing the task migration approach on a specific hardware. In this setup, the SpeedDreams simulator offers the virtual environment for some autonomous driving tasks, and the boards execute the tasks on GenodeOS. This serves as a basis for the setup to be implemented in this thesis as explained in the next section.

Although all these projects have achieved research-relevant results in their segments, more proof is needed to determine whether machine-learning-based migration planning can be adapted to different hardware and software configurations. Such proof is necessary before investigating the use of this technology on real vehicles. Therefore, the work in this thesis shall extend this research by testing the capabilities of the developments in combination with different technologies, such as neuromorphic hardware and the NRP.

Description of the Approach

As mentioned above, in this master's thesis, I aim to develop a Hardware-in-the-Loop setup for a model car or robot, using the NeuroRobotics Platform, embedded hardware and a neuromorphic platform, which will be running a machine learning algorithm for migration planning. This would allow for a new task distribution in a given failure scenario for one of the devices in the system. The NRP will provide the virtual environment for the car and the embedded hardware shall mimic a car to show its capabilities in a real situation.

The proposed Hardware-in-the-Loop setup consists of a virtual domain and a real, embedded domain, which interact with each other through the exchange of important information, such as sensor data and control commands. The virtual domain includes the NRP scenario, along with the ROS[4] topics / services for the interaction and the Gazebo model for the robot; this stage emulates the sensors and actuators as interfaces with the environment. The embedded domain refers to the model car architecture described below, running on embedded devices with FreeRTOS; these devices are responsible for executing the autonomous driving tasks, such as pathfinding, reading signals from the sensors, generating signals for motor control, and some dummy tasks to keep system utilization high. The utilization is artificially kept high to test the setup under critical situations, which could cause task sets to be unschedulable and deadlines to be missed, which should be mitigated by planning a new task distribution. Additionally, to perform the planning, a SpiNNaker neuromorphic board executes the machine learning algortithm for the schedulability analysis.

The NRP scenario shall present a pathfinding task for the robot in a labyrinth. This will be further extended by allowing for the connection with the real embedded hardware mimicking the car. This connection shall enable the exchange of the relevant sensor and control signals and will be implemented by ROS topics / services. ROS is chosen because it is already used in the NRP and Gazebo, and because it offers a simple way for communicating the virtual environment with the embedded devices, potentially reducing the development time for this stage.

The model car architecture consists of various embedded boards running on FreeRTOS, which will execute the afore mentioned tasks with real-time constraints. The path planning task shall be implemented, along with the reading of sensors and generation of control signals. Dummy tasks, on the other hand will be provided by COBRA framework, along with their execution times.

To complete the work within the envisaged time frame, certain parts of the machine learning approach shall be taken from previous works, especially the training data and some ideas for the neural network. The training data that has been previously generated in those projects shall be reutilized as much as possible, and for this purpose the dummy tasks shall be generated with the COBRA framework and shall resemble the ones utilized in MaLSAMi. Since the machine learning algorithms have not been implemented on the SpiNNaker board yet, a feed-forward net will be the initial approach, because it can be transformed to run on the neuromorphic board.

The proposed Hardware-in-the-Loop setup allows for fulfilling the goal of the thesis — determining whether the machine-learning-based approach for schedulability analysis is adaptable to new platforms and performs decently. To build it, following stages must be accomplished, each subdivided in tasks:

1. Building of an NRP scenario for a robot with an application for path planning.

   The NRP scenario is cloned from an existing scenario, which places a virtual robot in a labyrinth. Taking this as a starting point, the path planning algorithm

[4]ROS - Robot Operating System, website: https://www.ros.org/

shall be implemented, so that the robot detects the distance to the walls in the labyrinth and searches for an exit. Here, the important tasks to be considered are ensuring that the robot configuration allows for the correct navigation of the environment and a first version of the path planning algorithm in Python.

2. Integration of the model car in the NeuroRobotics Platform as the real robot. This way, the HIL setup can run the tasks in the embedded devices.

   The virtual robot in the NRP shall communicate with the model car by means of ROS topics / services. These shall allow the embedded devices in the model car to execute the path planning tasks based on the emulated sensor signals received and to send control signals to the virtual robot. The tasks to be considered in this stage are establishing the communication of both parts of the setup, the definition of the type of interfaces to be used (topics or services) and their implementation on the NRP simulation and on the embedded hardware.

3. Porting of the application in 1. to the real model car, so it can be executed on the embedded board, and development of additional dummy applications to ensure system utilization will be high to critical.

   The application developed in 1. shall be executed on real devices running on FreeRTOS. For this purpose, the application must be rewritten to C++ code and extended to use the ROS topics or services developed in 2. for reading the sensors and controlling the robot. The dummy tasks shall be generated using COBRA framework and basic FreeRTOS tasks, to resemble the tasks used in MaLSAMi. Using COBRA framework, we can also generate task sets with Worst-Case-Execution-Times, which will be used later as input for the machine learning algorithm.

4. Creation of an application failure scenario using the hybrid simulation setup (NRP + model car).

   A scenario shall be created to make the system fail in fulfilling all real-time constraints for the task set under the original task distribution. The failure scenario must ensure that the system is brought to its limits, so following are possible scenarios: a) suddenly shutting down one of the embedded boards, b) firing additional dummy tasks to force an utilization greater than 1 for one of the boards. After entering either of these scenarios, the system should only be able to meet all constraints by redistributing the tasks to the devices. Relevant tasks in this stage are the decision and implementation of the strategy for causing the failure, as well as ensuring that the system is brought to a critical state.

5. Mitigation of the application failure / recovery of the system using task re-deployment based on migration planning and machine learning performed on the SpiNNaker neuromorphic board

Based on previous work, a machine learning approach for schedulability analysis shall be implemented on the SpiNNaker board. Therefore, the first task is to create a feed-forward net that can be transformed to the neuromorphic board for this use case. Then the connection between the boards and the migration planning algorithm should be established. Afterwards, the planned distribution shall be achieved by redeploying the tasks onto the devices that are still available. Here, it shall be ensured that a feasible task distribution is selected.

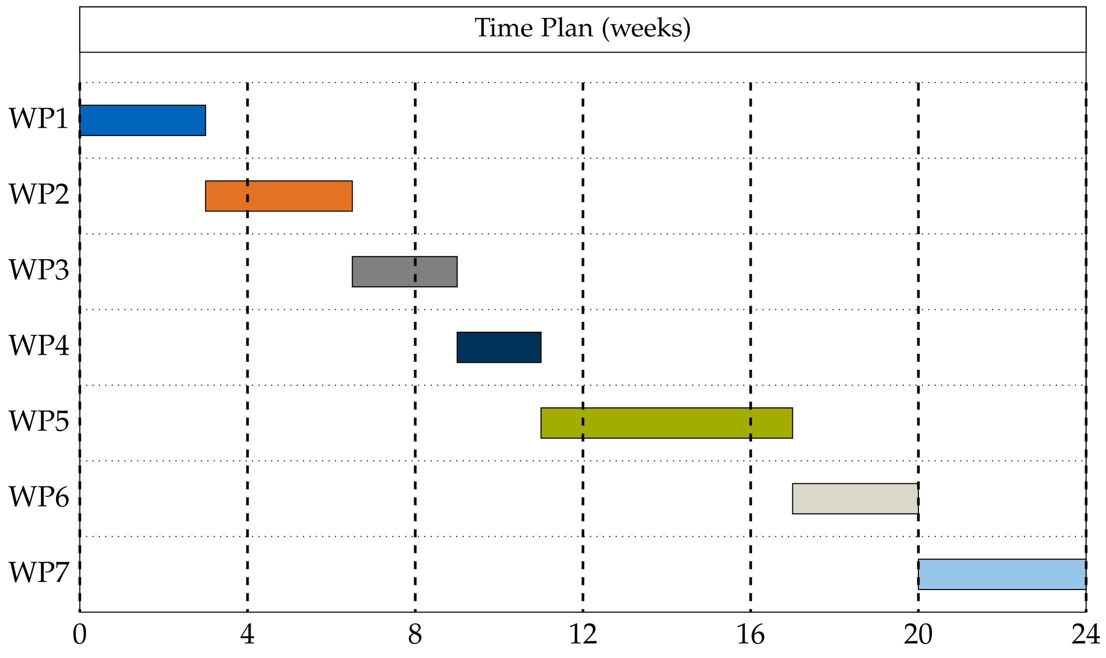
6. Timing evaluation of the results

   With hardware or software tools, the time between system failure and deployment of a feasible task configuration shall be measured. For this, time consumed in intermediate steps (such as the task distribution planning and the deployment) might be measured using End-to-End measurement. The results of these measurements shall be compared then to ones from previous works, such as those by Utz [Utz19] and Blieninger [Bli19].

This stage and task definition permits the fulfillment of the goal of the master's thesis in the planned time, as detailed in the time plan below.

Time Plan

The work will be organized in the work packages described below, whose relation to the stages described before is indicated in parenthesis. Some work packages are divided in subpackages for a more detailed description of the work comprised in them.



WP1. Initial setup of the existing independent components in the full system and familiarization with previous work and the algorithms and architectures used (3 weeks)

   a. Familiarization with previous work and other literature (1.5 weeks)
   b. Setup of the software tools to be used and familiarization with them (1.5 weeks)

WP2. Setup of initial NRP scenario (1.) and embedded hardware, along with their integration (2.) (3.5 weeks)

   a. Creation of initial NRP scenario and simple application (1 week)
   b. Creation of ROS topics and services that will serve as interface with model car (0.5 weeks)
   c. Setup of embedded hardware (1 week)
   d. Integration of model car hardware with NRP scenario using the topics and services created (1 week)

WP3. Development of the applications to be run on the hardware (3.) (2.5 weeks)

   a. Development of path planning application (1.5 weeks)
   b. Development of dummy applications to ensure critical system utilization (1 week)

WP4. Creation of application failure scenario (4.) (2 weeks)

WP5. Setup and integration of SpiNNaker board for performing the schedulability analysis and planning the task distribution (5.) (6 weeks)

   a. Setup and integration of SpiNNaker board (1.5 weeks)
   b. Implementation of the machine learning algorithm (2.5 weeks)
   c. Integration and implementation of the task distribution strategy (2 weeks)

WP6. Collection and analysis of the results (5.) (3 weeks)

   a. Refinement of HIL setup to show that the system can recover from the failure scenario implemented using the task distribution strategy (1.5 weeks)
   b. Timing analysis of the results (6.) (1.5 weeks)

WP7. Conclusion of the work and writing of the thesis (4 weeks)

Necessary Software and Hardware Components

In order to fulfill the purpose of the master's thesis in the planned time, I expect to need following materials:

- Setup with several embedded systems with FreeRTOS
- Work station for machine learning before test execution
- Work station for running NeuroRobotics Platform
- SpiNNaker board for executing machine learning at runtime

Bibliography

[Bli19] B. Blieninger. *Online Machine Learning for improved decision making regarding the migration of automotive software components at runtime*. 2019.

[BMJ18] G. Bersuker, M. Mason, and K. L. Jones. "Neuromorphic Computing: The Potential for High-Performance Processing in Space." In: *Game Changer* (Dec. 2018), pp. 1–12.

[Bur+18] O. Burkacky, J. Deichmann, G. Doll, and C. Knochenauer. *Rethinking Car Software and Electronics Architecture*. Tech. rep. McKinsey & Company, Feb. 2018.

[But11] G. C. Buttazzo. *Hard Real-Time Computing Systems*. 2011. DOI: 10.1007/978-1-4614-0676-1.

[EKB15] S. Eckl, D. Krefft, and U. Baumgarten. "KIA4SM - Cooperative Integration Architecture for Future Smart Mobility Solutions." In: *Conference on Future Automotive Technology (CoFAT)*. Vol. 4. Apr. 2015.

[Mun+17] P. Mundhenk, G. Tibba, L. Zhang, F. Reimann, D. Roy, and S. Chakraborty. "Dynamic platforms for uncertainty management in future automotive E/E architectures." In: *2017 54th ACM/EDAC/IEEE Design Automation Conference (DAC)*. 2017, pp. 1–6. DOI: 10.1145/3061639.3072950.

[NMM19] N. Navet, T. L. Mai, and J. Migge. *Using Machine Learning to Speed Up the Design Space Exploration of Ethernet TSN networks*. Tech. rep. University of Luxembourg, Jan. 2019.

[Tai19] S. Taieb. *Deep Learning Based Schedulability Analysis for Migration of Software Components at Runtime*. 2019.

[Utz19] T. Utz. *Vergleich von heuristischen und Deep Learning basierten Schedulability Analyse Verfahren für die Migration von Softwarekomponenten zur Laufzeit*. 2019.

[Vip+14] K. Vipin, S. Shreejith, S. A. Fahmy, and A. Easwaran. "Mapping Time-Critical Safety-Critical Cyber Physical Systems to Hybrid FPGAs." In: *2014 IEEE International Conference on Cyber-Physical Systems, Networks, and Applications*. 2014, pp. 31–36. DOI: 10.1109/CPSNA.2014.14.

[Vip18] K. Vipin. "CANNoC: An open-source NoC architecture for ECU consolidation." In: *2018 IEEE 61st International Midwest Symposium on Circuits and Systems (MWSCAS)*. 2018, pp. 940–943. DOI: 10.1109/MWSCAS.2018.8624006.